ERDC/ITL TR-19-3

Information Technology Laboratory

US Army Corps of Engineers®
Engineer Research and Development Center

ERDC
ENGINEER RESEARCH & DEVELOPMENT CENTER

Engineered Resilient Systems

TradeAnalyzer

Timothy W. Garton, Willie H. Brown, Eric R. Mixon,
and Joshua Q. Church

May 2019

The U.S. Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at http://acwc.sdp.sirsi.net/client/default.

Engineered Resilient Systems

TradeAnalyzer

Timothy W. Garton, Willie H. Brown, Eric R. Mixon,
and Joshua Q. Church

U.S. Army Engineer Research and Development Center (ERDC)
Information Technology Laboratory (ITL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Final Report

Prepared for Headquarters, U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under Engineered Resilient Systems Program, Data Analytics Work Package,
Collaborative Tradespace Analytics Work Unit 92L5D8

Abstract

Engineered Resilient Systems (ERS) is a program focused on transforming the methods and practices of designing and acquiring technologies of the future for the Department of Defense (DoD). There are areas of the program that specialize in design, execution, and analysis. All three areas are necessary for designing concepts, executing large sweeping simulations, and visualizing datasets in a manner that has never been available on datasets before.

The analysis area of ERS is an ever-developing program forming processes and methods to visualize exponentially growing sizes of data. ERS has developed an application called TradeAnalyzer that provides multiple visualization methods to examine, sort, and filter attributes of data. Scripting is provided to allow endless options for custom analytics. Select tools are provided for decision support to compare and contrast small sets of data once the dataset has been narrowed down to designs of interest that support the objective of the mission.

All of these methods combine to create a tool for evaluating alternatives and supporting the decision needs of the DoD in support of a more effective acquisition process.

Contents

Preface

This report is a deliverable product under the Engineered Resilient Systems (ERS) Program, Data Analytics Work Package, Collaborative Tradespace Analytics Work Unit 92L5D8. Dr. Owen J. Eslinger was the Program Manager, and Dr. Robert M. Wallace was the lead Technical Director of the ERS program.

The work was performed by the Scientific Software Branch (SSB) and Computational Analysis Branch (CAB) of the Computational Science and Engineering Division (CSED), Engineer Research and Development Center (ERDC), Information Technology Laboratory (ITL), Vicksburg, MS.

At the time of publication, Mr. Timothy W. Dunaway was Chief, SSB, Dr. Jeffrey L. Hensley was Chief, CAB, and Dr. Jerrell R. Ballard was Chief, CSED. The Deputy Director of ITL was Ms. Patti S. Duett and the Director was Dr. David R. Horner.

COL Ivan P. Beckman was the Commander of ERDC, and Dr. David W. Pittman was the Director.

Acronyms and Abbreviations

Term	Meaning
2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
ARL	Applied Research Laboratory at Pennsylvania State University
ATSV	ARL Trade Space Visualizer
CAB	Computational Analysis Branch
CAC	Common Access Card
CPU	Central Processing Unit
CSED	Computational Science and Engineering Division
CSV	Comma Separated Values
DoD	Department of Defense
DOM	Document Object Model
ERDC	U.S. Army Corps of Engineers, Engineer Research and Development Center
ERS	Engineered Resilient Systems
ES6	ECMAScript 6
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format 5
HPC	High Performance Computing
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IE11	Internet Explorer 11
ITL	U.S. Army Corps of Engineers, Engineer Research and Development Center, Information Technology Laboratory
JS	JavaScript
JSON	JavaScript Object Notation
LDV	Lightweight Database Virtualization
LTS	Long-Term Support
MI	Mutual Information

Term	Meaning
NPM	Node Package Manager
OS	Operating System
PHP	PHP: Hypertext Preprocessor
PING	PingFederate
RDE	Research and Development Environment
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
SAS	Statistical Analysis System
SELinux	Security-Enhanced Linux
SBD	Set-Based Design
SSB	Scientific Software Branch
USACE	U.S. Army Corps of Engineers
SSO	Single Sign On
WSTAT	Whole System Trade Analysis Tool

1 Introduction

1.1 Purpose: analyzing large data

Analyzing large data is the process of methodically and systematically making decisions to reduce incomprehensible datasets down to a manageable size that can be viewed and understood easily. These reductions are made by performing data analytics, producing metrics, identifying patterns, and/or producing any other criteria of comparison that can be mathematically modeled.

With the use of High Performance Computing (HPC) and the ever-shrinking limitations to storage and processing, the size of data that analytics can be performed on has been growing exponentially. Datasets that originally took years to manually process can now be interpreted and visualized in days or weeks.

TradeAnalyzer enables the user to readily analyze data through multiple methods. Custom R-scripts and objective functions can be applied to the tradespace in order to create additional metrics or reduce the tradespace to a more manageable size for future assessments. A tradespace is defined as the set of system and program parameters, properties, and facets needed to meet specific performance standards (Brantley et al. 2002). Additional information about a tradespace will be discussed in the section two.

Custom R-scripts are the primary technique of analytics in the tool suite. Mathematical models, or machine learning algorithms written in R, can be applied to the tradespace and executed to create plots, metrics, or reduce the tradespace for further analysis.

Objective functions can be written to link the interaction between the data and goals and used to reduce the tradespace for further analysis.

1.2 Background: trade set design analysis

TradeAnalyzer enables the user to readily analyze their data through multiple methods. Custom R-scripts and objective functions can be applied to the tradespace in order to create additional metrics or reduce the tradespace to a more manageable size for future assessments. A

tradespace is defined as the set of system and program parameters, properties, and facets needed to meet specific performance standards (Brantley et al. 2002). Additional information about a tradespace will be discussed in the section two.

Custom R-scripts are the primary technique of analytics in the tool suite. Mathematical models, or machine learning algorithms written in R, can be applied to the tradespace and executed to create plots, metrics, or reduce the tradespace for further analysis.

Objective functions can be written to link the interaction between the data and goals and used to reduce the tradespace for further analysis.

1.3 Objectives: visualizing large data

The implementation of set-based design (SBD) and the use of HPC are two related pillars of Engineered Resilient Systems (ERS) that require large tradespace visualization. The coupling of SBD and HPC generates large amounts of complex data that surpass conventional desktop-based data processing methods. As opposed to point-based design where the design cycle begins with a single design, SBD begins with a sufficient number of design alternatives to cover the feasible tradespace (Singer and Buckley 2009). This allows for analytical tradeoffs between system attributes and competing requirements. Hundreds of thousands of design-alternatives are often necessary to fully cover the tradespace. Generating these large datasets are often beyond the capacity of the standard desktop computers, thus, HPC is integral to the ERS workflow. The data output from the HPC machines is often too large to be ingested into typical desktop programs. In addition to technology constraints, the computer science expertise and time required to comprehend this data are typically outside the purview of most potential users.

TradeAnalyzer enables the user to readily explore large tradespaces using a variety of interactive visualizations, including histograms, two-dimensional (2D) and three-dimensional (3D) scatterplots, mutual information (MI) diagrams, parallel coordinate plots, and platform alternative scoring diagrams. These capabilities enable the user to explore multi-dimensional tradespaces. Additionally, visualizations can be rendered using custom R-scripts. The scripts can help to detect linear and non-linear relationships across attributes, find feasible clusters, and identify optimal platform solutions. Lastly, users can filter the data to

optimal solutions and compare them across many attributes. TradeAnalyzer provides the user increased computational power and visualization capability to support the acquisition decision-making process.

1.4 Approach: server rendering

Delivering data visualizations quickly and with real-time interactivity is critical. Since the TradeAnalyzer interface and client-side visualizations are done in JavaScript (JS), the user is limited in the amount of data that can be consumed and manipulated in the browser (Ziegler 2012). After approximately 100,000 records of a small address object, the interface begins to hang and the browser becomes unstable. Transferring large data from the server to the client machine also presents a bandwidth bottleneck that is avoided by processing the data on the server and returning only the results. TradeAnalyzer employs graphical processing units (GPUs) for mathematical calculations. GPUs are up to 20 times faster than the central processing units (CPUs) running in a browser (Varhol 2010). However, GPUs are more expensive to purchase and are only necessary when performing calculations on the data. TradeAnalyzer enables all of the users to utilize the GPU resources of one centralized server instead of having multiple systems with GPUs. Server rendering addresses the issues of data size, bandwidth, and CPU constraints by pooling common GPU resources on TradeAnalyzer servers. However, users must maintain constant internet connection to the server.

1.5 Scope: package and data management

Large data generation and various analytic methods necessitates the need to generate a process and format for both package and data management. TradeAnalyzer creates packages that store the data necessary for visualizations, along with the analytics that may have been run on the tradespace this package is stored in a Hierarchical Data Format 5 (HDF5) file, this is the standard convention for storage of large non-relational data due to its robust nature of dynamic data support. Tradespaces are often dynamic and require various columns, metadata, and space for storage that could not be defined easily in most other traditional formats.

In TradeAnalyzer, the package stores all attributes of the tradespace. The structure supports the storage of the raw data along with pre-processed binned data for quicker visualization generation. This combination of data

allows for quicker access and allows users to readily present the necessary visualizations. Users also have access to the underlying raw data for additional custom analytics.

1.6 Organization of this report

This report is organized into the following six chapters:

- Chapter 1 represents an overview of the situation at hand and the purpose of the report.
- Chapter 2 introduces the definition of a tradespace and discusses the industry standard analysis tools.
- Chapter 3 covers the structure of TradeAnalyzer and mechanisms used to assist and enhance analysis of large datasets.
- Chapter 4 provides the structure of how the system is implemented.
- Chapter 5 summarizes the report and application uses.
- Chapter 6 covers the approach for the future work and research of additional TradeAnalyzer tool suites.

2 Introduction to Tradespace Tools and Standards

This section defines a tradespace and covers the industry tools considered at the time of TradeAnalyzer development.

2.1 Tradespace

A tradespace is defined as the set of system and program parameters, properties, and facets needed to meet specific performance standards (Brantley et al. 2002). Tradespace can also be classified as the possible solution space extended across completely enumerated variables (Ross et al. 2004). The reasoning behind the tradespace enumeration process is to deter designers from narrowing their focus to point-based designs, allowing them to observe improved design solutions (Jilla et al. 2000).

2.2 Analysis tools

Many industry standard tools exist that have been reviewed for functionality, and limitations identified that necessitate the need to develop a custom Department of Defense (DoD) solution for tradespace analytics – TradeAnalyzer.

2.2.1 Whole System Trades Analysis Tool (WSTAT)

The Whole System Trades Analysis Tool (WSTAT) is a desktop decision support tool developed by Sandia National Laboratories and applied in partnership with the U.S. Army Program Executive Office for Ground Combat Systems. WSTAT incorporates various models into an integrated system view. This tool offers genetic algorithms to solve multi-objective optimization problems. WSTAT enables system design decision support along with multi-level tradespace exploration.

2.2.2 The Applied Research Laboratory at Pennsylvania State University (ARL) Trade Space Visualizer (ATSV)

The Applied Research Laboratory at Pennsylvania State University (ARL) Trade Space Visualizer (ATSV) is a desktop data visualization application developed in the Java programming language. This tool is capable of enabling users to visualize and explore multi-dimensional data. This tool is

licensed as an open-source application, allowing other developers to contribute to the program.

2.2.3 JMP

JMP is a proprietary data analysis desktop application developed by Statistical Analysis System (SAS). JMP enables users to quickly visualize and analysis data across different statistical tools. JMP offers a variety of tools such as data acquisition, “what-if” analysis, and data visualization.

2.2.4 TradeAnalyzer

TradeAnalyzer is a tradespace analysis and decision support tool. The previously mentioned analysis tools above are developed as desktop applications. Most potential DoD customers do not possess machines capable of utilizing the full potential of these tools due to the limited processing and computational power. This was the primary motivation behind the development of TradeAnalyzer – enabling the user to leverage a centralized server’s processing power to better support the DoD acquisition process. Additional information about TradeAnalyzer will be discussed in section three.

3 TradeAnalyzer

This section explains the features within TradeAnalyzer, along with the reasoning behind the decisions made during the development of the software.

3.1 Interface

Interface inspiration has come from a combination of industry standard analytic methods, along with newly researched mechanisms for portraying data visualization to the user.

3.1.1 Visualize/analyze data

Numerous tools have been created for the purpose of visualizing and analyzing tradespace data. Although these tools each serve a unique purpose, they often talk with each other in order to maximize usefulness.

The following are the tools that are available for use in TradeAnalyzer.

3.1.1.1 Histogram inspector

The histogram inspector visualizes each parameter and range and distribution of values. This allows a user to see detailed information about each parameter at a glance.

The histogram inspector was created so users can quickly select parameters that will be valuable for their analysis. Some parameters, for instance, may have no valuable information for the specific analysis being performed, and thus, may be excluded. Being able to quickly identify parameters allows users to deselect those parameters before moving forward.

3.1.1.2 Mutual information (MI) diagrams (analyze)

The MI diagram shows mutual information formulas of parameters and their relationships with other parameters. In it, parameters are separated into bins and the relationships between those bins are visualized.

MI diagrams were created in order to satisfy the need for users to see mutual information, a concept common in probability and information theory, between two or more variables. In contrast to correlation, which

identifies linear relationships, mutual information identifies both linear and non-linear relationships.

3.1.1.3 Parallel coordinates (compare)

Parallel coordinates allow users to see individual relationships between parameters, and particularly, the relationships between the distribution of values for one parameter and the distribution of values for another parameter. The parallel coordinates tool was created in response to the need to show individual relationships between two variables with a graph like representation of the values associated with the relationships.

3.1.1.4 Plot

The Plot tool allows the visualization of *X*, *Y*, and *Z* coordinates as either a 2D or 3D plot, and comes with a variety of tools to assist exploration and visualizing the data associated with these points. The Plot tool has the advantage of letting users visualize up to six dimensions, including three spatial dimensions, color, size, and opacity. It also allows for analysis of individual points of data, and is suitable for large number of data, close to 40 million rows at a time.

3.1.1.5 Explore

The Explore tool uses Plotly graphs to efficiently visualize the correlation of data between variables. There are various types of maps to explore, this allows for large amounts of visualization with minimal user effort. This tool enables users to explore a multitude of visualization options.

3.1.1.6 Examine

The Examine tool places individual points into scatterplots by parameters and allows for selections, coloring, comparisons using objectives, and the ability to save points. It was created to visualize and analyze individual points, allowing a critical analysis of data.

3.1.1.7 Select

The Select tool allows a user to define objectives, and compare multiple points based on these objectives, to identify which would best suit the objectives specified, and see a detailed comparison of one point versus another. This tool provides valuable insight that allows a user to see data

in various points and how they compare to each other and the users' objectives.

3.1.1.8 Combine

The Combine tool combines various tools together on one screen. Currently it enables the user to combine the Histogram Inspector, MI Diagrams, 3D scatterplots, and Parallel Coordinates, while also allowing access to: Package Export, Objective Coordinates, Workbench Layout, Annotations Editor, and Parameter/Field Selector. The combine tool allows users to view the relationships within the tools as various selections are made.

3.1.2 Filtering data

3.1.2.1 The field/parameter selection

The parameter selector allows the user to select and deselect certain parameters within the diagram to display only the results containing those parameters. It also provides a small histogram with the minimum and maximum values associated with the parameter. This data filter was created to help the user reduce the number of parameters seen in a toolset, which can help users isolate variables and relationships.

3.1.2.2 Objective function builder

The objective function builder allows creation and visualization of objectives and exporting of new subsets of the tradespace. Once an annotation has been created, it will operate as a field in the different tools.

3.1.2.3 Annotation editor

The annotation editor allows users to rate points based on desired field values. Once annotations are created, they can be combined into objective functions in the objective function builder.

3.1.3 R-Scripts (customize)

R-scripts were created to allow a user to add custom functionality to a tradespace by uploading scripts written in the R programming language. These scripts can reduce data sets to a more manageable and appropriate

datasets. It enables users to generate custom results from their datasets using previously written code.

3.1.4 Snapshots

Snapshots were created to enable users to easily retrieve previously selected data within the Customize and Select tools. Once a user creates a snapshot, the user can access the snapshot from the tradespace page. For example, if a user had selected parameters, defined objectives, and run visualizations, the user could save the snapshot and return to the tool with those parameters still selected, objectives defined, and visualizations rendered. This allows a user to save a snapshot of an analysis for future use.

3.2 Web-based

Limitations identified in desktop applications, from computing power to storage limitations and collaboration, have been rectified with Web-based hosting technologies.

3.2.1 Collaboration

Many DoD acquisition programs involve team members working in different geographic locations. Collaboration between those team members is essential, and because TradeAnalyzer is web-based, collaboration is straightforward. When a user creates a project in TradeAnalyzer, that user is the owner by default, and that user can add others with varying levels of permissions. Each user on a project has access to the data within that project. TradeAnalyzer contains a Snapshot function that enables a user to save the current state, including defined objectives, visualizations, and analytics. Everyone on the project can access that information as well. This enables users from various locations to access the same data and data manipulations from separate accounts.

3.2.2 Ease of access

The hurdle of geographic and technologic diversity among the users of TradeAnalyzer seemed best solved by using web technologies that are widely used and distributed. Therefore, TradeAnalyzer was created as a web application to facilitate ease of access for users without being limited by location, operating system, or type of machine used. As a web application, TradeAnalyzer has the potential to be accessed worldwide, by

any machine capability of browsing the Internet, and can be spread to the widest number of users without limiting factors.

3.2.3 Access to technology

Access to supercomputers may not be available to everyone, as building and maintaining a HPC is costly. A motivation of designing TradeStudio as a web-based tool is to give users access to resources not normally available. When dealing with tradespaces, technological capabilities are an important factor. Visualization and analysis of tradespaces require large amounts of storage space and computational power. In order to accurately capture some tradespaces, the technology needed can be on the scale of supercomputers (Stepanchick 2016).

3.2.4 Security

Security is always at odds with access. The more access a system has, the less secure it is, and vice-versa. The previously stated desires of collaboration, ease of access, and access to technology raise a difficult security challenge. The optimal way to balance security and access is to use a web server along with modern security practices to ensure data security (Fonseca et al. 2014).

One benefit of web-based technology from a security standpoint is enforcing operational security (Demchenko et al. 2005). A web environment can ensure proper procedures, such as preventing a user's unlocked computer from being accessed, and proper auditing. Ensuring complete security is impossible, as misuse by trusted users is always a threat, but helping prevent honest users from unintentionally leaking data is more possible in a web environment.

3.2.5 Authentication and authorization

Authentication and authorization are important aspects of security. Authentication is verifying the identity of a person, and authorization is allowing a person access to the appropriate information. By using two-factor authentication, with Common Access Card (CAC), or Yubikey for instance, TradeAnalyzer can ensure proper identification over the web. Once a user is properly authenticated, authorization can be enforced using proper access controls. While these methods can be utilized in a local environment, a web environment enforces the rules using technology.

3.3 Open, modular software architecture

One of the driving forces of the desired DoD custom software is to utilize open, modular software architectures in order to reduce cost of licensing, and utilize the growing list of tools by created by individuals in the open source profession.

3.3.1 Modular

Modular design is a hallmark of modern software engineering. Designing your software with several self-contained pieces allows for reuse, testing, and additions without having to recreate the entire project. By putting an emphasis on this design philosophy, TradeAnalyzer can change functionality and technologies easier than a monolithic product.

The modular approach is evident in many parts of TradeAnalyzer. The Representational State Transfer (REST) server has separate endpoints that all work independently, allowing endpoints to be both tested independently and added without affecting the other endpoints. The front-end code is written in Angular, which divides the JS and Hypertext Markup Language (HTML) into discrete, reusable components. The back end is broken up into separate servers, all with separate responsibilities, this allows for interchanging and upgrading one server without significant changes to other servers.

3.3.2 Open-source tools

The open-source software movement is vital when using web-based technologies. Open-source software provides large contributions from around the globe, allowing software to be patched and improved much more nimbly than by enterprise software.

Much of TradeAnalyzer's software is open-source, such as Apache, NodeJS, Python, MongoDB, AngularJS, and RedHat Linux. Google is the driving force behind the development of AngularJS. Even such software that is developed and maintained by enterprise business is open in nature and allows for more secure code as a community of users finds and reports vulnerabilities. The company Kitware, who helps develop TradeAnalyzer, has much of their code open source (i.e., graphic program ParaView and the web toolkit ParaViewWeb).

4 Infrastructure

The requirement that TradeAnalyzer must be web-based, defines industry standard and implementation best practices that have been reviewed in order to provide a safe and secure environment for data storage, access, and hosting.

4.1.1 Red Hat Enterprise Linux (RHEL) Server

All services for TradeAnalyzer run on Red Hat Enterprise Linux (RHEL) servers, also known as CentOS. RHEL is an open-source platform that provides professional support by paid subscription. RHEL provides security through Security Enhanced Linux (SELinux), ensuring that all processes only perform within their specified security context (Drepper 2004; Linux Redhat nd). The TradeAnalyzer developers also configured other security measures such as firewalls, system-level permission control, and isolation of functions on different machines.

When considering operating system (OS) options, other long-term support (LTS) Linux releases could have been sufficient, but the primary decision for choosing RHEL is the requirement that the OSs have provided support. The TradeAnalyzer software is also compatible with Windows Servers, however, it is both more expensive and requires more work to setup and maintain without any added Windows support or security benefits. This makes the Windows options undesirable. RHEL is run as a virtual machine inside of a managed server, and TradeAnalyzer does not employ any other dedicated hardware for any of its servers. This limits the cost of the computing services for ERS (Liu 2014).

4.1.2 Apache

Apache is used for the production Hypertext Transfer Protocol (HTTP) web server and serves several purposes. First, it hosts client-side application code (also referred to as static content) and delivers it to the user's browser upon request. Second, it performs redirect requests for other services to their proper endpoints. TradeAnalyzer has to filter out requests meant for the Application Programming Interface (API)-server and forward them along and filter out socket messages sent to the Kitware lightweight database virtualization (LDV) server. Lastly, TradeAnalyzer uses Apache to enforce authentication of requests through Ping tokens.

There are many HTTP web servers, Apache is one of the most commonly used that is authorized on DoD networks (Kumar et al. 2016). Apache is relatively easy to setup and configure for ERS purposes and provides all the needed services including serving static content and port redirects for API-server calls (proxy). The proxy functionality helps to protect against security threats that would attempt to circumvent Ping authorization as well as cross-site server scripting attacks.

4.1.3 HTML / JavaScript (ES6)

The TradeAnalyzer web client is programmed in HTML and JS. No additional plugins or special tools are necessary. The tools are fully supported in Chrome, Firefox, or Internet Explorer 11+ (IE11+) browsers. The JS code is also written for the future work of web development. TradeAnalyzer utilizes a JS module binder called Webpack, this allows for use of EMACScript 6 (ES6) features that are not yet fully supported in all modern browsers.

When choosing the language of a front-end web application, the primary industry alternatives are Hypertext Processor (PHP) or JS. Most other options are various ways to develop JS without actually writing it directly by transpiling other languages to it (Roby et al 2016). Transpiling takes the source code from one language and converts it into another. Since transpilers can easily fall behind in support for latest standards of JS or lose support altogether, the TradeAnalyzer development team chose to utilize some key frameworks such as AngularJS and Webpack to write JS directly. Maintaining a consistent JS environment made for easier workflow in the development process and enabled Node Package Manager (NPM) usage to manage and sync packages on the front and back end. The decision against using PHP was made to avoid the possible lapse in support for the transpilers and to keep a more consistent codebase.

4.1.4 Webpack

Webpack is an open-source JS module bundler. The combination of Webpack and NPM enables straightforward package management, HTML/JS minifying, and enforcement of code styling and standards, including improved cross-browser support when using ES6 capabilities (Roby et al. 2016). The build process is configured to use many tools in conjunction with Webpack to help support these features. This enables the

maintenance of a higher standard of code quality both now and throughout the development process.

Webpack on the front-end runs the code through a build process that condenses the web application to one JS file (Roby et al. 2016). This cuts down on the load time of the web page by reducing overhead of requesting dozens of pages when the user first visits the site. The build process also artificially gives support for some of the latest ES6 standards that are not yet supported on some browsers, such as IE11. This allows the TradeAnalyzer team to write code that extends the time window needing to be updated to keep up with the ever-changing world of JS. The team also runs code through ESLint, this ensures that all code in the application uses the same style and standards. This provides a consistent codebase where code in any part of the tool developed by any one team member will conform to those established style and standards. Having this consistency makes for easier understanding and updating by current and new code developers.

4.1.5 PingFederate

PingFederate (PING) Identity is the Single Sign On (SSO) service used by many applications on the Research and Development Environment (RDE) network. TradeAnalyzer utilizes PING for authentication. Authentication methods include CAC and username/password handshakes. This provides a secure way for the web application to verify the identity of each user (Pingidentity n.d).

PING protects TradeAnalyzer by taking initial requests and redirecting them through the authorization process on the PING server. A user attempts to visit TradeAnalyzer and is prompted for their CAC or username/password credentials. The PING server authenticates the user against Active Directory and forwards the user's request to TradeAnalyzer with the user's token appended in the header. This verifies the identity users hitting the server and can immediately proceed to the authorization security step when the request gets to the server.

4.1.6 AngularJS

AngularJS is an open-source JS framework [7, 8, 15] that drives the TradeAnalyzer development process on the client-side. AngularJS allows the TradeAnalyzer development team write more modular, reusable, and

structured JS code (Roby et al. 2016; Nilesh et al. 2015). While all of the tools could be written without AngularJS, the speed and quality and development are improved with the use of a good framework (Roby et al. 2016). AngularJS's data-binding does particularly well at reducing the amount of code used for manipulating the Document Object Model (DOM), and helps the development team focus its effort on code that makes the application work (Nilesh et al. 2015). There are other similar frameworks (React, AngularJS 2) that have been released since AngularJS 1.x.x, but upgrading to every new tool is too costly and time-consuming. Version 1.x.x of the AngularJS framework will be supported long term and will be modern and usable for the foreseeable future (Angularis n.d.).

AngularJS has a few primary constructs that the team utilizes to organize the front-end application code. These structures are Services, Controllers, and Directives. Services give all the tools of the application a consistent way to retrieve and sync data between themselves and the API-server. Every TradeAnalyzer tool uses the same suite of AngularJS services to interface with the backend API-server services. The directives in AngularJS enable easy means of writing very modular and reusable components. Even if the tool was drastically redesigned, the team could easily reuse many of the directives already written. Controllers are similar to directives, they are simply components without a strictly defined HTML template. Controllers are used as a consistent interface with a data model similar to a class in object-oriented programming languages (Angularis n.d.).

4.1.7 NodeJS

NodeJS is an open-source utility for writing server-side JS code (Node JS n.d.). Its primary purpose is to serve as a Representational State Transfer (RESTful) Web-API server. JavaScript is easy to write, and is the primary language of frontend web-tools. NodeJS development allows for consistent code since the frontend and backend are written in one language. NodeJS is also open-source and has a massive community of developers creating useful packages that make each task less burdensome (Node JS n.d.). For these reasons, the TradeAnalyzer team uses NodeJS to run TradeAnalyzer's API-server. The TradeAnalyzer web server is based on the NodeJS package, called Restify, this is a trimmed down version of ExpressJS. The API-server is the main gateway to server-side process, serving up projects, tradespaces, and images, while ensuring authorization for sensitive data. The API-server is the secured RESTful endpoints that drive all the functions of the web application. Documentation and libraries have been provided that allow for

other developers to utilize the same backend services in other tools if necessary. NodeJS is also fast and scalable to 1000's of consecutive users (Bangare et al. 2016).

NodeJS can run on virtually any OS and can be developed using anything from simple text editors to fully featured integrated development environments (IDE's) (Node JS n.d.). This makes the TradeAnalyzer backend tools easy to setup, test, and develop on practically any machine, without the use of costly and time-consuming virtual environments. Having an environment that doesn't need proprietary setup is also flexible. If new regulations change the requirements for the server, the software can run it, it should not take much of any effort to transition from one environment to another. Of all the tools that TradeAnalyzer uses, NodeJS is the most critical and would require the most effort to replace.

4.1.8 Node Package Manager (NPM)

The NPM comes natively with the installation of NodeJS. NPM is tool for managing project dependencies. There are dozens of other similar tools for almost any language, Nuget, Grunt, Bower, pip, etc. All libraries necessary for the API-server and also the front-end application for TradeAnalyzer are tracked in a JavaScript Object Notation (JSON) file and can be updated and maintained with one command.

By utilizing package management tools, the team avoids the time-consuming effort of manually installing each dependency whenever a new local or production environment is set up. With one command, the team can ensure that the correct version of all necessary libraries will be installed and ready to use. This also makes it easy to lock in specific versions of tools that might have new breaking changes and ensure that TradeAnalyzer has the latest non-breaking versions until development updates can be made. Although TradeAnalyzer does not directly need NPM to run, the developer benefits of utilizing this tool cannot be overstated.

4.1.9 MongoDB

MongoDB is the open-source, NoSQL database application that TradeAnalyzer uses to store project and user information (MongoDB n.d.). The information stored in MongoDB ties permission of users to package data, and stores users' application states. Package data is never directly stored in MongoDB, it is stored separately in HDF5 format. MongoDB

allows TradeAnalyzer to directly store JS objects as documents in the database. This lends itself to work well with NodeJS, a JS framework. NoSQL databases are also generally less vulnerable to many common attack vectors that compromise other databases (Győrödi et al. 2015). Although this does not remove all vulnerabilities, it is easier for the TradeAnalyzer team to protect and use (Okan et al. 2011).

MongoDB stores data in what is referred to as "Documents". These Documents are flexible and reduce much of the complexity that comes with designing a traditional relational database (Győrödi et al. 2015). TradeAnalyzer can handle frequent changes to the database model without having to perform time intensive efforts of exporting the database, dropping the tables, and rebuilding and re-importing with various scripts. The Document simply accepts the data that it is given. The more dynamic nature of the NoSQL database makes it desirable for the rapidly evolving features in TradeAnalyzer. Currently, there is not much data stored behind the API-server as the bulk of the information is stored in the tradespace HDF5 files. The MongoDB database tracks user information and permissions as well as basic information about projects, scripts, and tradespaces.

4.1.10 ParaView

ParaView is an open-source visualization program created by Kitware, Inc. TradeAnalyzer uses ParaView to visualize and analyze large data in many different ways. ParaView provides server-side visualizations when the data is too large to visualize in the browser, and for pre-computed statistics for client-side visualizations. ParaView also provides server-side Python scripts that aid in converting Comma Separated Values (CSV) datasets into HDF5 format with pre-computed values. This enables greater capability to work with larger data sets and still provide fast and efficient data visualizations.

ParaView makes up the vast majority of TradeAnalyzer's visualization tools and is currently critical for its usability. Like most of the tools in the suite, Kitware's visualizations are written in very modular fashion. This means that drastic redesign of the front-end interface could happen with little or no effort to reuse these visualizations. ParaView relies on C, C++, and Python as its underlying technologies.

4.1.11 Python

Python is one of the underlying languages of the ParaView tool. It is primarily a language used for scientific data science visualization and analytics. Currently, it is only required in TradeAnalyzer for running the ParaView tools. Python has become a popular language among data scientists and is especially suited toward script-based data visualization and analytics. It provides many pre-built modules that give the user a wealth of functionality to build from, without any drastic initial implementation.

Python is critical for the functionality of the ParaView technologies that make up the majority of TradeAnalyzer's visualizations. As it relates to applicability for TradeAnalyzer, Python is one of the few technologies that would be considered irreplaceable.

4.1.12 Hierarchical Data Format 5 (HDF5)

All tradespace data in TradeAnalyzer is stored on the backend in Hierarchical Data Format 5 (HDF5) format, even if it was uploaded as a CSV file. This file format enables the capability to store pre-computed analytics alongside the data and provides for fast and efficient data reading. There is a plethora of libraries for reading and manipulating HDF5 format in most languages making it effective and flexible.

HDF5 is a popular choice for data storage as it is compact, efficient, and supported in most languages (Folk et al. 2011). Many similar formats were considered, but HDF5 offered better performance and support. Currently, the format is deeply integrated with all parts of the TradeAnalyzer back end components, including ParaView, the API-server, and DeployR.

4.1.13 DeployR

DeployR is an Enterprise level web API-server that allows users to upload scripts written in R and execute them using server-side resources (Microsoft n.d.). DeployR has been recently bought by Microsoft and is therefore, a Microsoft Product. The features of DeployR help TradeAnalyzer to quickly provide users with the ability to write custom analytics scripts to run on their data within TradeAnalyzer.

DeployR is being rapidly updated and changed by Microsoft. The latest update introduces a complete redesign of the service endpoints that DeployR provides and would require a complete rewrite of the API-server's interface with it to support. Currently, TradeAnalyzer uses an older version without the new changes. Due to the limited nature of its tools, and the amount of work required to keep up with Microsoft changes, this application will likely be replaced in the near future.

5 Summary

In summary, ERS has supported the development of tools to analyze large datasets that have never been assembled, analyzed, or accessed before. TradeAnalyzer enables the user to readily analyze data through multiple methods using one centralized processing server hosted at ERDC. Unlike other analytical tools (i.e., JMP, WSTAT, or ATSV), TradeAnalyzer allows users to access a multitude of analytical tools through their web browser. Visualizations and analytic methods have been defined and provided in a way that will enhance and direct DoD analysis of future acquisitions trades decisions.

This report has served to identify the situation in process, provides a description of the structure of the system, and the method of implementation in the interface to support the need for tradespace analysis.

6 Future Work

Lessons have been learned in the creation of a web-based tool in support of the DoD need of data analytic tool suites. One of these lessons that will be researched and implemented in future work is the need for offline tools. This is similar to the desktop applications that are industry standard, but the technology will be implemented in a manner that is agnostic to the local or web-based platform in order to facilitate large scale HPC implementation, while also allowing for offline local implementation, if there is a necessity. The offline method will re-introduce the processing and storage limitations that cannot be rectified with local DoD computing.

6.1 Desktop edition

TradeAnalyzer enables users to readily analyze and visualize data through a connection to the TradeAnalyzer online server. In some scenarios, DoD customers do not have the means to connect to internet-based servers due to connection issues, project clearance requirements, etc. In order to mitigate these issues, the TradeAnalyzer developers will begin research into future development towards an offline desktop application. This application will leverage many of the same core capabilities as the current online edition.

6.1.1 Programming languages

The programming languages Python and R have grown in the data science and machine learning communities. Python is a general-purpose language, capable of quickly and efficiently performing data analytics. R is a statistical computing and graphics programming language built with the intension for statistical analysis. The current direction for TradeAnalyzer's desktop edition is to utilize both languages to support the ever-growing communities.

6.1.2 Interface

Python offers many ways to build interfaces for desktop applications but are generally complicated to develop. R offers "Shiny" applications, these provide a web framework capable of running locally in the user's browser. Web development interfaces offer a variety flexibility and customizability, so the aim is to leverage web technologies to handle the interface of the application.

6.1.3 Local server

In order to utilize web technologies locally on the desktop, a local server will be needed. The Python web frameworks Flask and Django are currently being considered as the handler for the local server. This local server would facilitate passing data from the interface to the Python or R scripts, and vice-versa. If R is used instead, the open-source package Shiny will handle the local server without any additional dependencies.

6.1.4 Visualizations

Both Python and R offer many open-source data visualizations libraries. They offer interactive and customizable visualizations options. Both libraries provide many out-of-the-box visualizations capable of quickly and easily representing data to the user.

6.1.5 Jupyter notebook

Jupyter Notebook is an open-source web application capable of performing data cleaning, statistical modeling, and data analysis. A notebook can contain live code, visualizations, text guides, and more, this can be shared easily amongst multiple users. Many DoD users need the flexibility to write fully customizable code for additional analysis purposes. The TradeAnalyzer desktop edition aims to enable users to access the notebooks feature within the application.

References

Angularjs. n.d..*AngularJS — Superheroic JavaScript MVW Framework*. Available at: https://angularjs.org/.

Bangare, S. L., S. Gupta, M. Dalal, and A. Inamdar. 2016. Using Node. Js to build high speed and scalable backend database server. In *International Journal of Research in Advent Technology Special Issue: National Conference NCPCI-2016*. http://www.ijrat.org/downloads/ncpci2016/ncpci-11.pdf.

Brantley, M. W., W. J. McFadden, M. J. Davis. 2002. Expanding the trade space: An snalysis of requirements tradeoffs affecting system design. *Acquisition Review Quarterly* Winter 2002.

Demchenko, Y., L. Gommans, C. de Laat, and B. Oudenaarde. 2005. Web services and grid security vulnerabilities and threats analysis and model. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing (GRID '05)*. Washington, DC: IEEE Computer Society 262–267. doi: https://doi.org/10.1109/GRID.2005.1542751.

Drepper, U. 2004. Security enhancements in redhat enterprise linux (beside selinux). *Security Elements in RHEL Version 1.2*. http://www.orkspace.net/secdocs/Unix/Protection/Description/Security%20Enhancements%20in%20Red%20Hat%20Enterprise%20Linux.pdf.

Folk, M., G. Heber, Q. Koziol, E. Pourmal, and D. Robinson. 2011. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, AD '11* 36–47.

Fonseca, J. N. Seixas, M. Vieira and H. Madeira. 2014. Analysis of field data on web security vulnerabilities. In *IEEE Transactions on Dependable and Secure Computing* 11(2): 89–100. doi: 10.1109/TDSC.2013.37.

Győrödi, C. R. Győrödi, G. Pecherle and A. Olah. 2015. A comparative study: MongoDB vs. MySQL In *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)* 1–6. doi: 10.1109/EMES.2015.7158433.

Jilla, C. D., D. W. Miller, and R. J. Sedwick. 2000. Application of multidisciplanary design optimization techniques to distributed satellite systems *Journal of Spacecraft and Rockets* 37(4): 481–490. https://doi.org/10.2514/2.3589.

Kumar, R., K. Subhash, and D. Sukanta. 2016. Adoption and evolution of FOSS: Key factors in the development of the Apache Web Server. *Recent Advances in Mathematics, Statistics and Computer Science*. 362–370. https://doi.org/10.1142/9789814704830_0033.

Linux Redhat. nd. *Red Hat Enterprise Linux operating system*. [online] Redhat.com. Accessed 18 Apr. 2017. https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux.

Liu, D., Y. Y. Zhang, N. Zhang, and K. Hu. 2014. A research on KVM-based virtualization security. *Applied Mechanics and Materials* 543–547. http://dx.doi.org.erdclibrary.idm.oclc.org/10.4028/www.scientific.net/AMM.543-547.3126.

Microsoft. n.d.. About DeployR - DeployR 8.x. https://msdn.microsoft.com/en-us/microsoft-r/deployr-about.

MongoDB. n.d.. *MongoDB for GIANT Ideas*. https://www.mongodb.com/.

Nilesh, J. Priyanka Mangal, and Deepak Mehta. 2015 AngularJS: A modern MVC framework in JavaScript. *Journal of Global Research in Computer Science* 5.12: 17–23.

Node JS. (n.d.). *Node.js*. https://nodejs.org/en/.

Okman, L., N. Gal-Oz, Y. Gonen, E. Guides, and J. Abramov.2011. Security issues in nosql databases. In *Trust, Security and Privacy in Computing and Communications (TrustCom) 2011 IEEE 10th International Conference on*. IEEE. doi: 10.1109/TrustCom.2011.70.

Pingidentity. n.d..Secure Access for the Digital Enterprise | Ping Identity. Available at: https://www.pingidentity.com/en.html (Accessed 19 Apr. 2017).

Roby, W. X. Wu, T. Goldina, E. Joliet, L. Ly, W. Mi, C. Wang, Lijun Zhang, D. Ciardi, and G. Dubois-Felsmann. 2016. Firefly: embracing future web technologies. In *Procedings Software and Cygerinfrastructure for Astronomy IV*. 9913. http://dx.doi.org/10.1117/12.22.33.042.

Ross, A. M., D. E. Hastings, N. P. Diller, and J. M. Warmkessel. 2004. Multi-attribute tradespace exploration as front end for effective space system design. *Journal of Spacecraft and Rockets* 41(1): 20–28). https://doi.org/10.2514/1.9204.

Singer, D., N. Doerry, and M. E. Buckley. 2009. What is set-based design. *Naval Engineers Journal* 121(4) 31–43. https://doi.org/10.1111/j.1559-3584.2009.00226.x.

Stepanchick, J. 2016. Integrating model based engineering and trade space exploration into naval acquisitions. Boston, MA: Massachusetts Institute of Technology. DSpace@MIT. http://hdl.handle.net/1721.1/104297.

Varhol, P. 2010. GPU vs. CPU computing. In *Engineering Computing: Digital Engineering*. http://www.digitaleng.news/de/gpu-vs-cpu-computing.

Zeigler, J. 2012. How big is too big for JSON. Josh Zeigler (blog) Web Development http://josh.zeigler.us/technology/web-development/how-big-is-too-big-for-json.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
May 2019	Final report	

4. TITLE AND SUBTITLE

Engineered Resilient Systems: TradeAnalyzer

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Timothy W. Garton, Willie H. Brown, Eric R. Mixon, and Joshua Q. Church

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
92L5D8

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Engineer Research and Development Center
Information Technology Laboratory
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

8. PERFORMING ORGANIZATION REPORT NUMBER

ERDC/ITL TR-19-3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Headquarters, U.S. Army Corps of Engineers
Washington, DC 20314-1000

10. SPONSOR/MONITOR'S ACRONYM(S)

USACE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Engineered Resilient Systems (ERS) is a program focused on transforming the methods and practices of designing and acquiring technologies of the future for the Department of Defense (DoD). There are areas of the program that specialize in design, execution, and analysis. All three areas are necessary for designing concepts, executing large sweeping simulations, and visualizing datasets in a manner that has never been available on datasets before.

The analysis area of ERS is an ever-developing program forming processes and methods to visualize exponentially growing sizes of data. ERS has developed an application called TradeAnalyzer that provides multiple visualization methods to examine, sort, and filter attributes of data. Scripting is provided to allow endless options for custom analytics. Select tools are provided for decision support to compare and contrast small sets of data once the dataset has been narrowed down to designs of interest that support the objective of the mission.

All of these methods combine to create a tool for evaluating alternatives and supporting the decision needs of the DoD in support of a more effective acquisition process.

15. SUBJECT TERMS

Engineering--Acquisition
Systems engineering
Big data
Quantitative research

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	35	